**Statistics impact on Query Plan**

Statistics play a significant role in query optimization and performance in database systems, including systems like Starburst, Trino, and traditional RDBMSs. They provide the query optimizer with essential information about data distribution, cardinality, and row counts. This data is used to generate an efficient query execution plan that minimizes resource usage and reduces query latency.

Here’s how statistics impact query plans:

**1. Join Selection and Ordering**

* **Join Algorithms**: Statistics help the optimizer choose the most efficient join algorithm (e.g., nested loop, hash join, or merge join) based on the expected number of rows and data distribution. For example, a hash join is typically preferred for large, non-indexed datasets, while a nested loop may work better for small tables.
* **Join Order**: The optimizer uses statistics to determine the best order for joins, often arranging the smallest tables first to reduce the overall data processing workload.

**2. Index Usage and Selection**

* **Selective Indexes**: If statistics indicate that a certain column is highly selective (e.g., it has many unique values), the optimizer may decide to use an index on that column, significantly speeding up data retrieval.
* **Index Skipping**: When statistics show a low selectivity (few distinct values), the optimizer might skip using an index altogether, as a full table scan could be faster.

**3. Predicate Pushdown**

* **Filtering Earlier**: Statistics can inform the optimizer to push down certain filters or conditions early in the query plan. For example, if a WHERE clause condition is highly selective, filtering data early reduces the volume of data that needs to be processed in later stages.
* **Partition Pruning**: If statistics show certain partitions contain most of the relevant data (e.g., date partitions in time-series data), the optimizer may skip irrelevant partitions, reducing I/O.

**4. Estimation of Memory and Resource Usage**

* **Memory Allocation**: Accurate statistics help the optimizer estimate how much memory will be needed for sorting, aggregation, and join operations. For instance, knowing the number of rows or the distribution of values helps in planning efficient memory allocation, avoiding overuse or shortages.
* **Parallel Execution and Distribution**: For distributed systems, statistics allow the optimizer to allocate tasks and balance load across nodes. This reduces data skew and resource bottlenecks by distributing data more evenly.

**5. Cost-Based Optimization (CBO)**

* **Cost Model Calculation**: Most modern optimizers, especially cost-based ones, rely on statistics to assign "costs" to different plan alternatives. Statistics such as row counts, distinct values, and distribution data enable the optimizer to pick the plan with the lowest estimated cost, balancing CPU, memory, and I/O requirements.
* **Avoiding Suboptimal Plans**: Without accurate statistics, the optimizer may choose suboptimal plans. For instance, it might perform unnecessary full table scans or make poor join and sorting choices, which can significantly increase query execution time.

**Example: Without Statistics vs. With Statistics**

* **Without Statistics**: Suppose the optimizer has no data on a table's row count or distribution. It might assume equal distribution across all values and choose a costly join or inefficient full scan, leading to high resource use.
* **With Statistics**: With statistics, the optimizer could choose a selective filter to minimize rows, use a faster join method based on row estimates, and skip unnecessary table scans, optimizing both performance and resource usage.

**Impact of Stale or Inaccurate Statistics**

* **Suboptimal Plans**: If statistics are outdated or inaccurate, the optimizer may generate a poor query plan, resulting in high latency and resource consumption.
* **Increased Maintenance**: Systems with high data churn (frequent updates or large data loads) may require frequent updates to statistics to maintain query performance.
* **Resource Misallocation**: Outdated statistics can lead to under- or over-provisioning of memory and CPU, causing potential errors, resource starvation, or inefficiency.

**Conclusion**

Effective query optimization heavily depends on up-to-date and accurate statistics. They allow the optimizer to select efficient paths for data processing, reduce I/O, and leverage the most appropriate join types and indexing strategies. Regularly updated statistics are critical to ensure that query plans remain efficient over time, particularly as data volume and distribution change.

tributed and large-scale data environments.